ISSN 1561-5359. Штучний інтелект, 2017, № 1

УДК 004.932

П.Ю. Сабельніков
Інститут кібернетики імені В.М. Глушкова НАН України, Україна
пр. Академіка Глушкова, 40, м. Київ, 03680

БАГАТОРІВНЕВЕ РОЗШАРУВАННЯ КОНТУРІВ ОБ'ЄКТІВ У БІНАРНИХ ЗОБРАЖЕННЯХ

P.Yu. Sabelnikov
V.M. Hlushkov Institute of Cybernetics of the NAS of Ukraine, Ukraine
40, Academician Hlushkov av., Kyiv, 03680

MULTILEVEL BUNDLE OF OBJECTS CONTOURS IN BINARY IMAGES

У статті пропонується модифікований алгоритм паралельного сканування і кодування контурів об'єктів у бінарних зображеннях з введенням процедур їх розшарування за рівнями. Алгоритм дозволить впорядкувати в ієрархічному порядку чорні і білі об'єкти зображень та надасть можливість обчислення моментів інерції об'єктів різного рівня, з включенням або виключенням деяких з них.

Ключові слова: зображення, контур, моменти інерції.

The article proposes a modified algorithm for parallel scanning and coding of object contours in binary images with the introduction of procedures for their layering by levels. The algorithm will allow you to order black and white objects of images in a hierarchical order and will provide an opportunity to calculate the moments of inertia of objects of various levels, with the inclusion or exclusion of some of them.

Key words: images, contour, moments of inertia.

Вступ

Достатньо важливою при обробці зображень, особливо вбудованими пристроями, що призначені для роботи в реальному часі, є проблема прискорення обчислень. В цілях скорочення часу запізнювання результатів обробки, економії ресурсів пам'яті та підвищення швидкодії за рахунок використання тільки внутрішньої швидкої пам'яті процесора, а також векторних операцій, автором раніше був запропонований алгоритм паралельного кодування контурів об'єктів у бінарних зображеннях [1].

Алгоритм заснований на методі послідовного сканування частини рядків зображення [2], що передбачає послідовний аналіз взаємного розташування чорних і білих сегментів в черговому та попередньому рядках. При цьому можуть виникати ситуації: зародження двох гілок контуру, з'єднання двох гілок контуру, перехід гілки з рядка в рядок.

В результаті аналізу всього бінарного зображення формуються вектори контурів всіх чорних і білих об'єктів, що об'єднують координати точок гілок, які належать кожному з контурів.

Запропонований в [1] алгоритм не дозволяє встановити зв'язок між об'єктами зображення.

В даній статті пропонується модифікований алгоритм паралельного кодування контурів, який дозволяє розшарувати та впорядкувати чорні і білі об'єкти бінарних зображень, що в результаті надасть додаткову інформацію про структуру бинарного зображення для подальших етапів обробки. Наприклад, при обчисленні моментів інерції об'єктів різного рівня, цілеспрямовано включати або виключати деякі з них.

Для прискорення процесу обчислень передбачається використання векторних операцій. Під векторними операціями маються на увазі інструкції, які дозволяють

обробляти велику кількість упакованих даних за одну операцію. Наприклад, на платформі x86 (фірма Intel) існує велика кількість SIMD розширень: MMX, 3DNow, SSE, SSE2, SSE3, SSEE3, SSE4.1, SSE4.2, AVX і AVX2. До переліку входять операції пересилання, упаковки, розпаковки даних, арифметичні і логічні операції, операції порівняння з формуванням вектору ознак, а також специфічні операції, наприклад BSF (Bit Scan Forvard) і BSR (Bit Scan Reverse) - визначення індексу значущої одиниці в коді праворуч і ліворуч. В процесорах ARM також реалізована підсистема векторних операцій для обробки чисел з фіксованою та плаваючою точкою (розширення Neon).

Модифікований алгоритм паралельного кодування контурів з багаторівневим розшаруванням об’єктів у бінарних зображеннях

Для повного розуміння пропонованих доповнень до алгоритму скористаємося наступним прийомом. По можливості, коротко повторюючи матеріал викладений автором в попередній статті [1], більш детально розглянемо доповнення. При цьому будуть виправлені деякі помилки допущені в зазначеній статті.

Отже розглядається бінарне зображення, в якому на білому фоні представлені чорні об’єкти (або навпаки, це не принципово), що мають замкнені контури і позначені, як об’єкти 1-ого рівня. В їх межах можуть бути розташовані білі об’єкти 2-ого рівня. В межах об’єктів 2-ого рівня можуть бути розташовані чорні об’єкти 3-ого рівня і так далі. Контур визначається, як замкнена ломана лінія товщиною в один піксель, яка з’єднує центри граничних пікселів об’єкту, що примикають друг до друга, по прямій або діагоналі і належать цьому об’єкту. Щоб не виникало колізій з перетинанням контурів встановлено, що точка між пікселями при переходах по діагоналі належить чорним об’єктам і є перепоною для білих об’єктів.

В якості приклада будемо розглядати двох рівневе бінарне зображення представлене на рисунку 1. Для коректної роботи алгоритму об’єкти не повинні стикатися з краями зображення, тому для довільного зображення крайні рядки і стовпці заповнюються фоном.

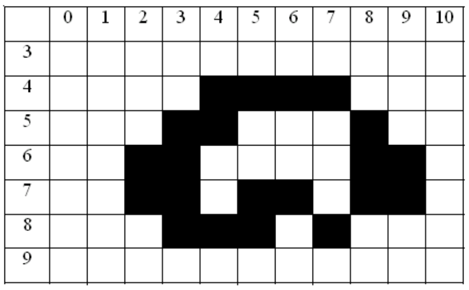


Рис. 1. Тестове бінарне зображення

Для формального представлення алгоритму введемо деякі поняття і позначення.

Поточний і результуючий рядки зображення будемо представляти векторами T і R, де кожній точці зображення відповідають 2 біта. Вектори VN та V_N містять номери гілок, що перетинають відповідно попередній і поточний рядки.

В векторі T формуємо бінарне зображення, наприклад 4-ого рядка.

T=00,00,00,00,01,01,01,01,00,00,00.

Здійснюємо перетворення $T = T \oplus (T\rangle\rangle 2)$, де

$\oplus$ - операція логічної нерівнозначності,

$\rangle\rangle 2$ - операція зсуву вектора на 2 біта вправо.

В результаті отримуємо T=00,00,00,00,01,00,00,00,01,00,00 де значущі позиції відповідають початкам чорних і білих сегментів зображення в рядку.

В вектор R поміщаємо вектор T, зсунутий на 1 біт вліво ($R = T\langle\langle 1$), і вважаємо його, як вектор сформований для попереднього рядка. Таким же чином формуємо в векторі T поточний 5-ий рядок та операцією логічного додавання $\cup$ додаємо його до вектора R. В результаті отримуємо злиті та впорядковані за номерами точок початки сегментів попереднього і поточного рядків.

R=00,00,00,01,10,01,00,00,11,01,00.

Надалі для простоти будемо представляти результуючі вектори не в бітовій, а в кодовій формі, де

00=0 означає, що в цій позиції немає початків сегментів,

01=1 означає початок сегменту в (i) - ому рядку,

10=2 означає початок сегменту в (i-1) - ому рядку,

11=3 означає одночасний початок сегментів в (i) - ому та в (i-1) - ому рядках.

Отже для 5-ого рядка R=0,0,0,1,2,1,0,0,3,1,0.

Початки сегментів кожного рядка обов'язково чергуються, за початком чорного сегменту йде початок білого сегменту, а за початком білого сегменту йде початок чорного сегменту. Тому при послідовному аналізі R буде відомо не тільки, в якому рядку (попередньому чи поточному) зустрівся початок сегменту, а і якого він кольору.

Таким чином алгоритм включає наступні дії.

На початку аналізу i та вектори R і VN у всіх позиціях дорівнюють 0.

1. Послідовно з (i) - ого рядка бінарного зображення формується і перетворюється вектор T.
2. $R = R \cup T$.
3. Аналіз R та VN і заповнення таблиці гілок контурів та вектора V_N.
4. $R = T\langle\langle 1$, VN= V_N, i=i+1.
5. Дії з п.1 повторюються доки не дійдемо до кінця зображення.
6. Об'єднання гілок в контури.

В результаті сукупність гілок контурів об'єктів буде представлена у вигляді таблиці гілок контурів 1, компонентами якої, є:

n – номери гілок (спочатку співпадають з порядковими номерами гілок, після закінчення аналізу будуть змінені на загальні номери контурів),

N +1 – загальна кількість гілок (буде відома в кінці аналізу),

p – ознака гілки (1 – чорна гілка, 0 – біла гілка),

y – номер (координата) рядка, з якого починається гілка,

z – кількість точок в гілці,

x – номери (координати) точок гілки в рядку,

n_p – номер гілки, з кінцем якої здійснюється з'єднання.

Додаткові компоненти таблиці 1:

n_c – номер найближчої гілки контуру, яка передує парі гілок, що зароджуються (при відсутності такої гілки записується спеціальний код).

На етапі 6 алгоритму вводиться додаткова таблиця контурів 2 з наступними параметрами:

kk – індексний номер контуру за рівнями (_._._. ...), наприклад kk=*2.1.4.0* означає, що об'єкт 3-ого рівня з номером 4 знаходиться в межах об'єкта 2-ого рівня з номером 1, який теж знаходиться в межах об'єкта 1-ого рівня з номером 2 (максимальна кількість можливих рівнів задається на початку, для наведеного прикладу становить 4);

r – рівень вкладеності контуру (чим більше *r* тим нижче рівень);

pk – ознака об’єкта (1 – чорний об’єкт, 0 – білий об’єкт);

sk – лічильник номерів внутрішніх контурів об’єктів, що на один рівень нижче по відношенню до поточного контура (в кінці аналізу буде дорівнювати кількості вкладених об’єктів);

k0 – номер контура об’єкта, що на один рівень вище по відношенню до поточного контура (при відсутності такого контура записується спеціальний код);

n0 – номер початкової гілки контуру в таблиці гілок контурів.

Таблиця 1. Таблиця гілок контурів

n_c	*n*	*p*	*y*	*z*	*x*	*n_p*
	0					
	1					
	2					
	3					
	………………					
	N					
	N+1					

Таблиця 2. Таблиця контурів

k k	*r*	*pk*	*sk*	*k0*	*n0*
…………					

Алгоритм аналізу і формування таблиці гілок контурів

Для пояснень процедури 3 «Аналізу *R* і заповнення таблиці гілок контурів» введемо позначення початків чорного і білого сегментів бінарного зображення:

1 – початок чорного сегменту (*i*-1) - ого рядка,

$\underline{1}$ – початок чорного сегменту (*i*) - ого рядка,

0 – початок білого сегменту (*i*-1) - ого рядка,

$\underline{0}$ – початок білого сегменту (*i*) - ого рядка.

Згідно з методом сканування рядків зображення можливі ситуації при послідовному аналізі, що відповідають наступним комбінаціям слідування початків сегментів за порядковими номерами точок попереднього і поточного рядків:

1. ($\underline{1}$, 1, …), (1, $\underline{1}$, …), ($\underline{0}$, 0, …), (0, $\underline{0}$, …) – продовження гілок з рядка в рядок;

2. (0, $\underline{\ldots}$, $\underline{\mathbf{1}}$, $\underline{\mathbf{0}}$, $\underline{\ldots}$, *), (1, $\underline{\ldots}$, $\underline{\mathbf{0}}$, $\underline{\mathbf{1}}$, $\underline{\ldots}$, *) – виділені жирним шрифтом комбінації відповідають зародженню пари з’єднаних початками гілок в поточному рядку (зірочка позначає перший не підкреслений індекс, що зустрінеться);

3. ($\underline{0}$, …, **1**, **0**, ..., $\underline{*}$), ($\underline{1}$, …, **0**, **1**, ..., $\underline{*}$) – виділені жирним шрифтом комбінації відповідають з’єднанню кінців пари гілок в попередньому рядку, що означає приписування до параметрів кожної гілки номеру гілки, з якою вона з’єднується (підкреслена зірочка позначає перший підкреслений індекс, що зустрінеться);

4. Для інших сполучень трьох індексів продовжити аналіз.

Згідно з умовою, що точка між чорними пікселями при переходах по діагоналі належить чорним об’єктам, при одночасній зустрічі в одній позиції комбінацій 0 і 1 першою вважається одиниця.

Алгоритм реалізовано за допомогою автоматної моделі, що циклічно, в залежності від значень окремих позицій вектора *R*, які подаються на вхід автомата, змінює свій стан та відповідно здійснюються необхідні дії. Таблиця переходів автомату представлена в таблиці 3.

Аналіз вектора *R* можна здійснювати послідовно позиція за позицією або за рахунок визначення значущих позицій, використовуючи наприклад раніше наведені операції BSR або BSF.

Стан автомату вказує, які дві попередні значущі позиції вектора *R* потрібно враховувати. Тобто, коли стан позначено (0, $\underline{0}$), з кодом 0, це означає, що в попередніх позиціях поточного і попереднього рядків були початки білих сегментів. Відповідно надалі в обох рядках можуть зустрітися тільки початки чорних сегментів.

В стовпчику «Дії при зміні стану» таблиці 3 однією зірочкою позначені дії при зародженні гілок, двома – при з'єднанні гілок, трьома – при переході гілки з рядка на рядок.

Таблиця 3. Таблиця переходів автомату з відповідними діями

Поточний стан та його код (*stan*)	Код входу (*vxid*)	Вхідна ознака	Наступний стан та його код	Дії при зміні стану
(0, $\underline{0}$), 0	0	----	(0, $\underline{0}$), 0	-------------
	1	$\underline{1}$	(0, $\underline{1}$), 1	*i_n=i_n+1, xx=j.*
	2	1	($\underline{0}$, 1), 2	*in=in+1.*
	3	1, $\underline{1}$	(1, $\underline{1}$), 4	*** *nn=vn[++in], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.*
(0, $\underline{1}$), 1	0	----	(0, $\underline{1}$), 1	-------------
	1	$\underline{0}$	(0, $\underline{0}$), 0	* *n_c[ss]= v_n[i_n-1],* *v_n[i_n++]=ss, n[ss]=ss, p[ss]=1, y[ss]=i, zz=0, x[ss][zz]=xx, z[ss++]=1,* *v_n[i_n]=ss, n[ss]=ss, p[ss]=0, y[ss]=i, zz=0, x[ss][zz]=j, z[ss++] =1.*
	2	1	($\underline{1}$, 1), 3	*** *nn=vn[++in], v_n[i_n]=nn, zz=z[nn], x[nn][zz]=xx, z[nn]=zz+1.*
	3	1, $\underline{0}$	(1, $\underline{0}$), 6	*** *nn=vn[++in], v_n[i_n++]=nn, zz=z[nn], x[nn][zz]=xx, z[nn]=zz+1,* xx=j.
($\underline{0}$, 1), 2	0	----	($\underline{0}$, 1), 2	-------------
	1	$\underline{1}$	(1, $\underline{1}$), 4	*** *nn=vn[in], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.*
	2	0	($\underline{0}$, 0), 5	** *nn=vn[in] , n_p[nn]= vn[in+1],* *nn=vn[in+1] , n_p[nn]= vn[in++].*
	3	$\underline{1}$, 0	($\underline{1}$, 0), 7	*** *nn=vn[in++], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.*
($\underline{1}$, 1), 3	0	----	($\underline{1}$, 1), 3	-------------
	1	$\underline{0}$	(1, $\underline{0}$), 6	*i_n=i_n+1, xx=j.*
	2	0	($\underline{1}$, 0), 7	*in=in+1.*
	3	0, $\underline{0}$	(0, $\underline{0}$), 0	*** *nn=vn[++in], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.*
(1, $\underline{1}$), 4	0	----	(1, $\underline{1}$), 4	-------------
	1	$\underline{0}$	(1, $\underline{0}$), 6	*i_n=i_n+1, xx=j.*
	2	0	($\underline{1}$, 0), 7	*in=in+1.*
	3	0, $\underline{0}$	(0, $\underline{0}$), 0	*** *nn=vn[++in], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.*
($\underline{0}$, 0), 5	0	----	($\underline{0}$, 0), 5	-------------
	1	$\underline{1}$	(0, $\underline{1}$), 1	*i_n=in+1, xx=j.*
	2	1	($\underline{0}$, 1), 2	*in=in+1.*
	3	1, $\underline{1}$	(1, $\underline{1}$), 4	*** *nn=vn[++in], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.*
(1, $\underline{0}$), 6	0	----	(1, $\underline{0}$), 6	-------------
	1	$\underline{1}$	(1, $\underline{1}$), 4	* *n_c[ss]= v_n[i_n-1],* *v_n[i_n++]=ss, n[ss]=ss, p[ss]=0, y[ss]=i, zz=0, x[ss][zz]=xx, z[ss++]=1,* *v_n[i_n]=ss, n[ss]=ss, p[ss]=1, y[ss]=i, zz=0, x[ss][zz]=j, z[ss++] =1.*
	2	0	($\underline{0}$, 0), 5	*** *nn=vn[++in], v_n[i_n]=nn, zz=z[nn], x[nn][zz]=xx, z[nn]=zz+1.*
	3	$\underline{1}$, 0	($\underline{1}$, 0), 7	* *n_c[ss]= v_n[i_n-1],* *v_n[i_n++]=ss, n[ss]=ss, p[ss]=0, y[ss]=i, zz=0, x[ss][zz]=xx, z[ss++]=1,* *v_n[i_n]=ss, n[ss]=ss, p[ss]=1, y[ss]=i, zz=0, x[ss][zz]=j, z[ss++] =1, in=in+1.*
($\underline{1}$, 0), 7	0	----	($\underline{1}$, 0), 7	-------------
	1	$\underline{0}$	(0, $\underline{0}$), 0	*** *nn=vn[in], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.*
	2	1	($\underline{1}$, 1), 3	** *nn=vn[in] , n_p[nn]= vn[in+1],* *nn=vn[in+1] , n_p[nn]= vn[in++].*
	3	1, $\underline{0}$	(1, $\underline{0}$), 6	** *nn=vn[in] , n_p[nn]= vn[in+1],* *nn=vn[in+1] , n_p[nn]= vn[in++], i_n=i_n+1, xx=j.*

В поясненнях задіяні такі проміжні змінні та вектори:

i – номер поточного рядка,

ss – лічильник номерів гілок (перед початком аналізу *ss=0,* в кінці аналізу буде дорівнювати кількості гілок),

stan – код стану автомата, *stan* =0 перед початком аналізу кожного вектора *R,*

j – номер поточної позиції вектора *R*, *j=0* перед початком аналізу кожного вектора *R*,

vxid – код значення поточної позиції вектора *R*,

nn – проміжне значення номеру гілки,

zz – проміжне значення кількості координат в гілці,

xx – проміжне значення номера значущої позиції вектора *R*.

VN – вектор номерів гілок, що перетинають попередній рядок, де *vn[in]* – окремі компоненти вектора з індексом *in, ++in* – операція перед індексації, *in++* – операція пост індексації,

V_N – вектор номерів гілок, що перетинають поточний рядок, де *v_n[i_n]* – окремі компоненти вектора з індексом *i_n*.

Для розшарування контурів об’єктів за рівнями в позиціях таблиці 3, що відповідають зародженню гілок, вводиться процедура запису в параметр *n_c* номера найближчої гілки контуру, яка передує парі гілок, що зароджуються. В подальшому на етапі 6 алгоритму буде визначено де знаходиться об’єкт, гілки якого зароджуються. Чи знаходиться він в межах контуру об’єкта, якому належить ця гілка або є об’єктом того ж рівня, тобто знаходиться в межах спільного для них контуру об’єкта більш вищого рівня.

В таблиці 4 відображені результуючі значення векторів *T, R, VN, V_N* та коментар до ситуацій, що виникли при обробці кожного з рядків зображення.

Таблиця 4. Результат обробки тестового зображення за рядками

i	T_i , R_i , *j=0-10*	*VN / V_N*	Коментар
3	*T* =00000000000 *R* =00000000000	_,	Дій не відбувається
4	*T* =00001000100 *R* =00001000100	_,0,1	Зародження пари гілок з номерами 0, 1
5	*T* =00010100110 *R* =00012100310	_,0,2,3,1	Перехід гілок 0, 1 з рядка в рядок Зародження пари гілок з номерами 2, 3
6	*T* =00101000101 *R* =00121200321	_,0,2,3,1	Перехід гілок 0, 2, 3, 1 з рядка в рядок
7	*T* =00101101101 *R* =00303101303	_,0,2,4,5,3,1	Перехід гілок 0, 2, 3, 1 з рядка в рядок Зародження пари гілок з номерами 4, 5
8	*T* =00010011100 *R* =00212213302	_,0,6,7,1	Перехід гілок 0, 1 з рядка в рядок З’єднання кінців гілок 2 і 4 Зародження пари гілок з номерами 6, 7 З’єднання кінців гілок 5 і 3
9	*T* =00000000000 *R* =00020022200	_,	З’єднання кінців гілок 0 і 6, 7 і 1

В таблиці 5 представлено вміст таблиці гілок контурів після аналізу кожного рядка, з чого достатньо легко можна дослідити його послідовну зміну. В рядку параметра *n_c* прочерком позначено те, що гілкам, які зароджуються, не передує ніяких інших гілок.

В таблиці 6, як приклад, надано детальний опис дій та значення змінних в процесі послідовного аналізу 8-ого рядка тестового зображення. В стовпчику позначеному *j* розташовані номери позицій вектора *R*, починаючі з першої значущої, а в стовпчику позначеному *vxid* значення цього вектора відповідно позицій. В стовпчиках позначених *stan, ss, in, i_n, xx, V_N* – значення цих змінних, які вони приймають по закінченню аналізу відповідної позиції. Вектор *V_N* послідовно заповнюється номерами гилок, що перейшли з попереднього, або зародилися в

поточному рядку. Опис дій надано у вигляді переліку операцій над змінними скопійованими з таблиці 3 відповідно значень *stan, vxid* та тих же операцій, але з підстановкою чисельних даних. Результат цих операцій, що змінили вміст таблиці контурів можна порівняти для контролю з відповідними значеннями таблиці 4 для *i*, що дорівнює 8.

Таблиця 5. Вміст таблиці гілок контурів у процесі обробки тестового зображення за рядками

n_c	*n*	*p*	*y*	*z*	*x*	*n_p*
	i=4					
-	0	1	4	1	4	
	1	0	4	1	8	
i=5						
-	0	1	4	2	4, 3	
	1	0	4	2	8, 9	
0	2	0	5	1	5	
	3	1	5	1	8	
i=6						
-	0	1	4	3	4, 3, 2	
	1	0	4	3	8, 9, 10	
0	2	0	5	2	5, 4	
	3	1	5	2	8, 8	
i=7						
-	0	1	4	4	4, 3, 2, 2	
	1	0	4	4	8, 9, 10, 10	
0	2	0	5	3	5, 4, 4	
	3	1	5	3	8, 8, 8	
2	4	1	7	1	5	
	5	0	7	1	7	

n_c	*n*	*p*	*y*	*z*	*x*	*n_p*
	i=8					
-	0	1	4	5	4, 3, 2, 2, 3	
	1	0	4	5	8, 9, 10, 10, 8	
0	2	0	5	3	5, 4, 4	0
	3	1	5	3	8, 8, 8	5
2	4	1	7	1	5	2
	5	0	7	1	7	3
0	6	0	8	1	6	
	7	1	8	1	7	
i=9						
-	0	1	4	5	4, 3, 2, 2, 3	6
	1	0	4	5	8, 9, 10, 10, 8	7
0	2	0	5	3	5, 4, 4	4
	3	1	5	3	8, 8, 8	5
2	4	1	7	1	5	2
	5	0	7	1	7	3
0	6	0	8	1	6	0
	7	1	8	1	7	1

Таблиця 6. Результати аналізу 8-ого рядка зображення (початок аналізу: *i=8; VN=_,0,2,4,5,3,1; ss=6; in=0, i_n=0*)

j	*stan*	*vxid*	*ss, in, i_n, xx* *V_N*	Дії при аналізі компонентів вектора *R*, сформованого для поточного рядка за його окремими значущими компонентами
2	0	2	*6, 0, 0, _* *V_N=_*	*stan=2, in=in+1* *in=0+1*
3	2	1	*6, 1, 0, _* *V_N=_,0*	**** stan =4,nn=vn[in], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.* *nn=0, v_n[1]=0, zz=z[0]=4, x[0][4]=3, z[0]=zz+1=5.*
4	4	2	*6, 1, 1, _* *V_N=_,0*	*stan =7, in=in+1=2.*
5	7	2	*6, 2, 1, _* *V_N=_,0*	*** stan =3, nn=vn[in] , n_p[nn]= vn[in+1], nn=vn[in+1] , n_p[nn]= vn[in++].* *nn=vn[2]=2 , n_p[2]= vn[3]=4, nn=vn[3]=4 , n_p[4]= vn[2++]=2.*
6	3	1	*6, 3, 1, _* *V_N=_,0*	*stan =6, i_n=i_n+1, xx=j.* *i_n=1+1, xx=6.*
7	6	3	*6, 3, 2, 6* *V_N=_,0,6,7*	** stan =7, n_c[ss]= v_n[i_n-1],* *v_n[i_n++]=ss, n[ss]=ss, p[ss]=0, y[ss]=i, zz=0, x[ss][zz]=xx, z[ss++] =1,* *v_n[i_n]=ss, n[ss]=ss, p[ss]=1, y[ss]=i, zz=0, x[ss][zz]=j, z[ss++] =1, in=in+1.* *n_c[6]= 0,* *v_n[2++]=6, n[6]=6, p[6]=0, y[0]=8, zz=0, x[6][0]=6, z[6++] =1,* *v_n[3]=7, n[7]=7, p[7]=1, y[7]=7, zz=0, x[7][0]=7, z[7++] =1, in=3+1=4.*
8	7	3	*8, 4, 3, 6* *V_N=_,0,6,7*	*** stan =6, nn=vn[in] , n_p[nn]= vn[in+1],* *nn=vn[in+1] , n_p[nn]= vn[in++], i_n=i_n+1, xx=j.* *nn=vn[4]=5 , n_p[5]= vn[5]=3,* *nn=vn[5]=3 , n_p[3]= vn[4++]=5, i_n=3+1, xx=8.*
9	6	0	*8, 5, 4, 8* *V_N=_,0,6,7*	*stan =6, дії не відбувається.*
10	6	2	*8, 5, 4, 8* *V_N=_,0,6,7,1*	*stan =5, nn=vn[++in], v_n[i_n]=nn, zz=z[nn], x[nn][zz]=xx, z[nn]=zz+1.* *nn=vn[++5]=1, v_n[4]=1, zz=z[1]=4, x[1][4]=8, z[1]=4+1.*

Об'єднання гілок у контури

На етапі 6 алгоритму виконується об'єднання гілок в контури, тобто пошук першої гілки контуру, визначення рівня вкладеності контуру *r* та індексного номера контуру *kk*, слідкування за гілками, що відносяться до контуру одного об'єкта, запис в параметр *n* всіх гілок контуру загального номера контуру *k* та перерахунок координат гілок.

Визначення рівня вкладеності контуру і номера контуру за рівнями базується на таких логічних висновках:

- Гілки контура верхнього рівня зароджуються раніше ніж вкладені в них гілки контурів нижнього рівня.
- Контури опрацьовуються послідовно в порядку їх зародження.
- Знайдена початкова гілка і об'єднані з нею в спільний замкнений контур інші гілки позначаються загальним номером контура і ігноруються при пошуку початкової гілки слідуючого контура. Тому наступна, обов'язково парна, гілка знайдена в порядку зародження гарантовано не є складовою раніше опрацьованих контурів і являється початковою гілкою контура. Відповідно її ознака *p* вкаже на колір об'єкта (чорний або білий), що буде замкнений контуром, початком якого є ця гілка.
- Гілка, номер якої на етапі 3 був записаний в параметр *n_c* при зародженні чергової початкової гілки, входить у вже опрацьований контур і всі його параметри визначені. В тому числі колір об'єкта, до складу контура якого входить ця гілка.

Процедура об'єднання гілок така.

1. Вводиться змінна загального лічильника контурів *k* та лічильника контурів першого рівня *k1*. На початку процедури *k* і *k1* дорівнють нулю в кінці аналізу будуть дорівнювати відповідно загальній кількості контурів і кількості контурів першого рівня.

2. Пошук початкової гілки контуру та заповнення таблиці контурів для знайденого контуру.

В таблиці гілок контурів скануються по вертикалі номери гілок. Гілка, номер якої більше або дорівнює *k* вважається початковою гілкою контура. Позначимо цей контур, як K1. В параметр *n0[k]* таблиці контурів записується порядковий номер знайденої гілки, а в параметр *pk[k]* ознака кольору цієї гілки, що є ознакою кольору об'єкта. Параметру *sk[k]* присвоюється значення 0.

В параметрі *n_c* початкової гілки контуру може знаходитись номер найближчої до неї гілки, що передувала їй при зародженні або спеціальний код в разі відсутності такої гілки. Останнє означає, що знайдений контур є контуром 1-ого рівня. В свою чергу, якщо така гілка є, в параметрі *n_c* цієї гілки знаходиться порядковий номер вже опрацьованого контуру, якому вона належить. Позначимо цей контур, як K0 при цьому параметр *k0[k] = n[n_c]* з таблиці гілок контурів.

В рядку таблиці контурів з номером *k* обчислюються і записуються значення параметрів контуру в залежності від можливих ситуацій:

- Знайдений контур є контуром 1-ого рівня. Тоді параметрам контуру K1 присвоюються значення: *k1=k1+1*, перший індекс *kk[k] = k1, r[k] =1, k0[k]=спеціальний символ* (означає, що контуру вищого рівня не має).
- Кольори об'єктів замкнених контурами K1 і K0 різні $pk[k] \neq pk[k0]$. Це означає, що K1 являється внутрішнім, тобто на 1 рівень нижче по відношенню до K0. При цьому параметрам контуру K1 присвоюються модифіковані значення параметрів контуру K0. Відповідно *r[k] = r[k0]+1, kk[k] = kk[k0]*. Потім параметр

sk[k0] контуру К0 збільшується на одиницю і індексу з номером *r[k]* індексного номера *kk[k]* контуру К1 присвоюється значення параметра *sk[k0]*.

- Кольори об'єктів замкнених контурами К1 і К0 однакові *pk[k] = pk[k0]*. Це означає, що К1 і К0 одного рівня. Якщо рівень *r[k0]* контуру К0 дорівнює одиниці, то для К1 виконуються дії, як для контуру першого рівня. Якщо, навпаки, параметру *k0[k]* контуру К1 присвоюється значення *k0[k0]* контуру К0 і здійснюються дії, як при умові *pk[k]* $\neq$ *pk[k0]*. Тобто контуром на 1 рівень вищим по відношенню до К1 буде той же контур, що і для К0.

3. Слідкування за гілками контуру.

Враховуючі, що при зародженні пара гілок з'єднана своїми початками, а на з'єднання кінців вказує значення параметра *n_p*, починається послідовне слідкування за гілками з'єднаними кінцями і початками, доти поки не повернемось до гілки, з якої почали. Номери з'єднаних між собою гілок *n* змінюються на значення *k*.

Загальна послідовність координат точок в контурі за гілками має наступний порядок. Для першої гілки береться послідовність координат від початку гілки до її кінця, для наступної, навпаки, від кінця до початку і так далі, по черзі змінюючи напрям. Цей порядок відповідає обходу контуру проти годинникової стрілки.

Колір об'єкту замкненого контуром (чорний або білий) визначає ознака першої гілки. Враховуючі умову, що контур утворюють граничні пікселі, які належать об'єкту, координати *x* всіх білих гілок чорного контуру, та координати *x* всіх чорних гілок білого контуру зменшують на одиницю.

4. Збільшуємо *k* на одиницю і продовжуємо процес сканування з пункта 2 процедури доки всі гілки таблиці гілок контурів не будуть опрацьовані.

В таблиці 7 представлена фінальна таблиця гілок контурів, а в таблиці 8 фінальна таблиця контурів після вказаних перетворень. В результаті отримані контури двох об'єктів, чорного і білого.

Таблиця 7. Фінальна таблиця гілок

n_c	*n*	*p*	*y*	*z*	*x*	*n_p*
-	0	1	4	5	4, 3, 2, 2, 3	6
	0	0	4	5	7, 8, 9, 9, 7	7
0	1	0	5	3	5, 4, 4	4
	1	1	5	3	7, 7, 7	5
2	1	1	7	1	4	2
	1	0	7	1	7	3
0	0	0	8	1	5	0
	0	1	8	1	7	1

Таблиця 8. Фінальна таблиця контурів

kk	*r*	*pk*	*sk*	*k0*	*n0*
1.0.0.0	1	1	1	-	0
1.1.00	2	0	0	0	2
....					

З таблиці 7 видно, що для горизонтальних ліній, відображені координати тільки їх крайніх точок. Процедура формування повного вектору координат точок контуру досить проста. Тому не будемо її детально розглядати оскільки для обчислення моментів інерції бінарних об'єктів вона не потрібна.

Обчислення моментів інерції об'єктів бінарних зображень

Одними з параметрів, що характеризують форму об'єктів, є моменти інерції різних порядків. На їх основі, для задач порівняння об'єктів за формою контурів, можна будувати моментні інваріанти, які не залежать від афінних перетворень: зсуву, повороту, зміни масштабу об'єкта та інших перетворень [3].

У роботі [1] представлено ефективний алгоритм розрахунку моментів інерції з використанням векторних операцій. Нижче наведений вираз для обчислення моментів довільного порядку:

$$M_{j,k-j} = \sum_{i=1}^{N} B_i \cdot x_i^j \cdot y_i^{k-j}, \; j = (0, k),$$ де:

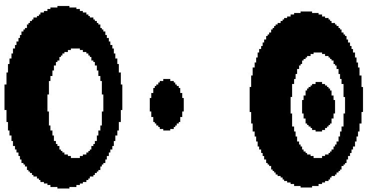
x_i^j і y_i^{k-j} – координати пікселів об'єкта,

k – максимальний порядок моментів інерції,

B_i – для бінарних об'єктів, замкнених контуром, дорівнює 1.

З виразу видно, що моменти складних багаторівневих об'єктів можна представляти як суму або різницю моментів окремих складових бінарного зображення. Щоб цілеспрямовано це робити, на подальших етапах обробки зображення необхідно мати інформацію про структуру і взаємозв'зки об'єктів на зображенні.

Наприклад, на рисунку 2 представлено два зображення. Необхідно обчислити момент в об'єктах першого рівня тільки чорних частин. Згідно з таблицею гілок контурів, яка для обох зображень буде відрізнятись тільки значеннями координат *x*, загальна кількість об'єктів дорівнює 3. Тобто, чорний, білий і ще один чорний об'єкт з моментами $M^1{}_{j,k-j}$, $M^2{}_{j,k-j}$, $M^3{}_{j,k-j}$, де верхній індекс є номером об'єкта. Відповідно, для кожного з зображень маємо два чорні об'єкти 1 і 3, з моментами $M^1{}_{j,k-j}$ і $M^3{}_{j,k-j}$.



а б

Рис. 2. Бінарні зображення

Розшарувавши об'єкти зображень та сформувавши таблицю контурів отримуємо наступну інформацію. На рис.2а маємо два об'єкти. Перший двох рівневий з моментами чорної частини ($M^1{}_{j,k-j}$-$M^2{}_{j,k-j}$), другий одно рівневий з моментами $M^3{}_{j,k-j}$. На рис. 2б маємо один трьох рівневий об'єкт з моментами чорної частини ($M^1{}_{j,k-j}$-$M^2{}_{j,k-j}+M^3{}_{j,k-j}$). Погляд на зображення і розрахунок моментів після розшарування значно змінились.

Висновки

В результаті досліджень запропоновано модифікований алгоритм виділення, кодування та розшарування об'єктів бінарних зображень, що дозволяє отримати додаткову інформацію про структуру зображення і взаємозв'язок його об'єктів.

Це надає можливість на подальших етапах обробки цілеспрямовано виділяти окремі суттєві об'єкти зображень або їх частини, обчислювати, наприклад, їх моменти інерції і моментні інваріанти, порівнювати з еталонами в задачах розпізнавання або контролю форми.

В роботі на конкретних прикладах продемонстровані принципи використання векторних операцій за запропонованими алгоритмами. На основі цих принципів можуть бути розроблені і інші більш ефективні алгоритми.

Коректність роботи алгоритму підтверджена шляхом порівняння результатів його роботи з результатами роботи алгоритму послідовного кодування контурів, реалізованого автором раніше.

Література

1. Сабельніков П.Ю. Паралельне кодування контурів об'єктів у бінарних зображеннях та обчислення їх моментів інерції / П.Ю. Сабельніков // Науково–теоретичний журнал «Штучний інтелект» (Київ) – 2016. – №2(72). – С. 35 – 45.
2. Абламейко С.В. Обработка изображений: технология, методы, применение: Учеб. пособие. / С.В. Абламейко, Д.М. Лагуновский. – М.: Амалфей, 2000. – 304 с.
3. Глумов Н.И. Построение и применение моментных инвариантов для обработки изображений в скользящем окне / Н.И. Глумов // Компьютерная оптика. – 1995. – № 14. – С. 46-54.

Literatura

1. Sabel'nikov P.Yu. Paralel'ne koduvannya konturiv ob"yektiv u binarnykh zobrazhennyakh ta obchyslennya yikh momentiv inertsiyi / P.Yu. Sabel'nikov // Naukovo–teoretychnyy zhurnal «Shtuchnyy intelekt» (Kyyiv) – 2016. – #2(72). – S. 35–45.
2. Ablameiko S.V. Obrabotka izobradzeni: tehnologiya, metodi, primenenie: Ucheb. Posobie. / S.V. Ablameiko, D.M. Lagunovski. – M: Amalffei, 2000. – 304 s.
3. Glumov N.I. Postroenie i primenenie momentnih invariantov dlya obrabotki izobradzeni v skolzyachem okne / N.I. Glumov // Komputernaya optika. – 1995. – № 14. – S. 46-54.

RESUME

P.Yu. Sabelnikov

Multilevel bundle of objects contours in binary images

In the processing of images, especially embedded devices designed for real-time operation, the problem of accelerating computation is important. To reduce the delay time of processing results and increase the speed by using only the internal fast processor memory, as well as vector operations, this article proposes a modified parallel contour coding algorithm. The algorithm allows you to layer and order black and white objects of binary images, as a result provides additional information about the structure of the binary image for subsequent processing steps. The obtained additional information will allow to calculate the moments of inertia of objects of different levels with the inclusion or exclusion of some of them. To accelerate the calculation process, the use of vector operations is assumed.

Vector operations are instructions that allow you to process a large number of packed data in one operation. For example, on the x86 platform (Intel company) there are a lot of SIMD extensions: MMX, 3DNow !, SSE, SSE2, SSE3, SSEE3, SSE4.1, SSE4.2, AVX and AVX2. In ARM processors, a subsystem of vector operations for processing numbers with fixed and floating point (Neon extension) is also implemented.

The algorithm for extracting and encoding loops is based on sequential introduction of image lines and analysis of the order of the passage in them of black and white parts of the image. To implement the algorithm in the memory of the processing device, it is necessary to store only the entered and the previous lines of the image. The result of the algorithm is two tables. The first table of branches of contours contains information on the number of branches and their common numbers in the order of origin, the coordinates of the points of branches of contours, and also the links of each branch to the next branch for the possibility of their binding into integral contours. The second table of contours contains data about the hierarchical level of the object, a characteristic (a black or white object), a reference to its initial branch in the branch table, and an index number indicating objects of higher levels to which it refers.

In the article on the concrete examples, we show how to use vector operations to implement the proposed algorithms. Based on these examples, other, more efficient algorithms can be developed.

Надійшла до редакції 10.06.2017